

Representation of Events in Nerve Nets and Finite Automata

Stephen C Kleene (1909-1994)

- Automata studies (1956)

Kleene's Regular Expressions

provides a language to program finite state automata, or to specify the behaviour of such machines.

Program execution is represented as a string of actions occurring sequentially while the program runs.

Operators and constants

- $p;q$ is sequential composition
 - execution of p is followed by execution of q
- $p \text{ u } q$ is choice between p and q
 - only one of them is executed
- **1** is executed by doing nothing and making no change

The Laws of Regular Algebra

- $p;(q;r) = (p;q);r$; associates and so does \mathbf{u}
- $\mathbf{1};p = p = p;\mathbf{1}$ $\mathbf{1}$ is the unit of ;
- $p \mathbf{u} q = q \mathbf{u} p$ \mathbf{u} commutes
- $p \mathbf{u} p = p$ \mathbf{u} is idempotent
- $p;(q \mathbf{u} r) = p;q \mathbf{u} p;r$; distributes thru \mathbf{u} and leftward as well

Refinement Ordering \leq (below)

- $p \leq q$ means every execution of p is an execution of q ,
so p is more determinate – stronger
and q is more abstract – weaker
- If p is a program and r, s are specifications
 $p \leq r$ means p satisfies r
 $r \leq s$ means r implies s
- The algebra ignores the distinctions, because the proofs are the same for both specifications and programs

$p \leq q$ means $p \mathbf{u} q = q$

Covariance

- Theorems:

$$p ; q \leq p ; (q \text{ u } r)$$

$$q ; p \leq (q \text{ u } r) ; p$$

- Proof rule:

$$\frac{p \leq q}{\begin{array}{l} p ; r \leq q ; r \\ r ; p \leq r ; q \end{array}}$$

If the antecedents above the line have been proved,
so are the consequences below the line.

More proof rules for \leq

$$\frac{p \leq q \quad \& \quad q \leq r}{p \leq r}$$

(\leq is transitive)

$$\frac{p \leq q \quad \& \quad q \leq p}{p = q}$$

(\leq is antisymmetric)

All the rules can be derived from the laws, by a little proof

1. Proofs of programs

An Axiomatic Basis for Computer Programming

C.A.R. Hoare

Communications of the ACM 12(10) 1969

The Hoare triple

- Purpose: to prove that all possible executions of a program q when started after a precursor p will exhibit some desirable property r .
- Definition: $\{p\} q \{r\} = p;q \leq r$
- Interpretation:
 - If p (the precondition) describes what has happened so far
 - and q is now started and executed to completion,
 - then the trace of overall execution will satisfy r (the postcondition).

Rule: Sequential composition

$$\frac{\{p\} q \{s\} \quad \& \quad \{s\} q' \{r\}}{\{p\} q;q' \{r\}} \quad (\text{Hoare})$$

It is equivalent to a weaker form of the law of associativity

$$p;(q;q') \leq (p;q);q' \quad (\text{Proof by covariance})$$

It is not possible from the rule to prove $(q';q);p \leq q';(q;p)$,

A Calculus of Communicating Systems

A.J.R.G Milner

Springer Lecture Notes in Computer Science

1980

Milner Transitions

- Purpose: to show how an implementation can generate just a single execution of the program r .
- Definition: $r \xrightarrow{q} p = q;p \leq r$
- Interpretation:
 - r may be executed by first executing q ,
 - with p as continuation for later execution.
 - (maybe there are other ways of executing r)

Rule: Sequential composition

$$\frac{r \xrightarrow{q'} s \quad \& \quad s \xrightarrow{q} p}{r \xrightarrow{q';q} p}$$

is equivalent to the other weak form of associativity

$$(q';q);p \leq q';(q;p),$$

By antisymmetry of \leq , conjunction of the Hoare inequality with the Milner inequality gives the associative equation.

Summary: Sequential Composition

Milner
transition

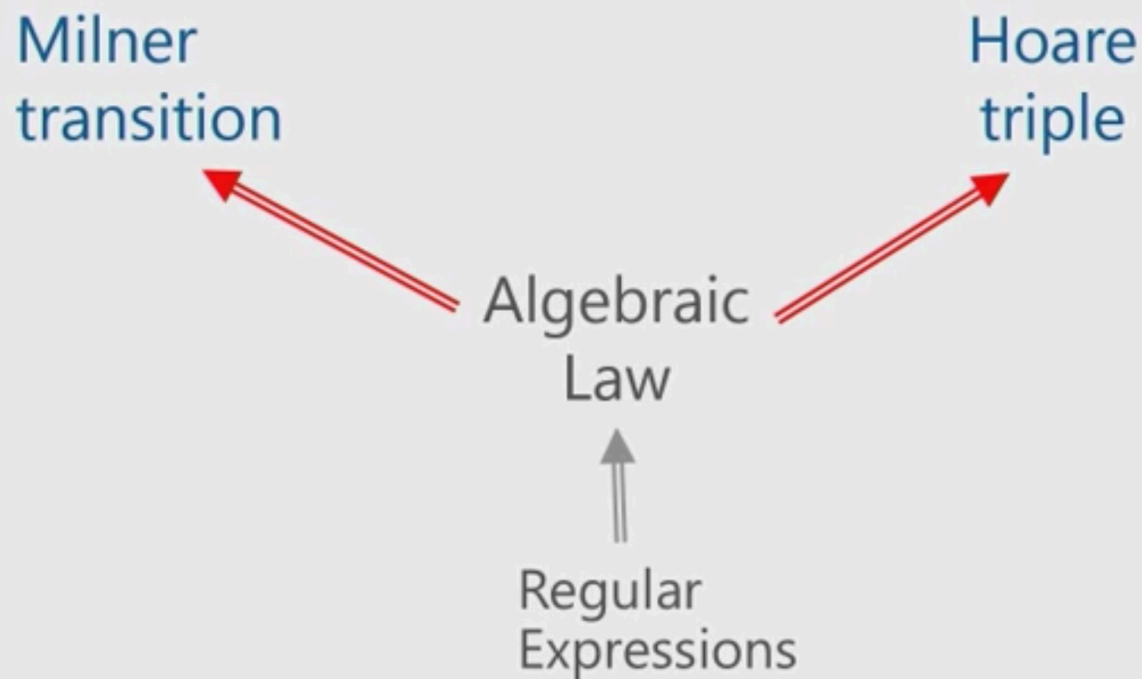
Hoare
triple

Algebraic
Law



```
graph TD; AL[Algebraic Law] ==> MT[Milner transition]; AL ==> HT[Hoare triple];
```

Summary: Sequential Composition



Concurrent Composition: $p||q$

- p and q start together and finish together
 - in between, they may communicate.
- $||$ is associative, **commutative**, and idempotent
- and distributes through **u**
- and has unit **1**
- and obeys the Exchange Law ...

The exchange law

- Axiom: $(p \parallel q) ; (p' \parallel q') \leq (p ; p') \parallel (q ; q')$
- The two $'$'s on the RHS of \leq are scheduled to occur simultaneously, as shown by the single $;$ on the LHS.
- LHS is a more interleaving sequential implementation of the more general concurrency of RHS
- The LHS is a subset of the interleavings of the RHS

The Exchange Law

- Axiom: $(p \parallel q) ; (p' \parallel q') \leq (p ; p') \parallel (q ; q')$
- Theorems (frame laws):
 1. $(p \parallel q) ; q' \leq p \parallel (q ; q')$
 2. $p ; (p' \parallel q') \leq (p ; p') \parallel q'$
 3. $p ; q' \leq p \parallel q'$ and $q ; p' \leq p' \parallel q$

Proof: substitute **1** for the variable(s) of the axiom that are omitted in the theorem

Interleaving example

- Let a, b, c, d be atomic actions of a sequential program
 - written without semicolons as $abcd$
- Let x, y, z, w be atomic actions of another such program
 - written $xyzw$
- The two programs are executed concurrently
- the next slide shows how the Exchange Law permits an interleaved execution of actions from the two sequences

Interleaving by exchange

$$\begin{aligned} & \text{abcd} \parallel \text{xyzw} \\ (\text{assoc } ;) &= (\text{a;bcd}) \parallel (\text{xy;zw}) \\ (\text{exchange}) &\geq (\text{a}\parallel\text{xy}) ; (\text{bcd}\parallel\text{zw}) \\ &= (\text{a}\parallel\text{x;y}) ; (\text{b;cd}\parallel\text{zw}) \\ (\text{frame}) &\geq (\text{a}\parallel\text{x}) ; \text{y} ; (\text{b}\parallel\text{zw}) ; \text{cd} \\ (\text{comm } \parallel \dots) &\geq \text{xayz}\text{bwcd} \end{aligned}$$

Different associations and commutations of $;$ and \parallel at each step will obtain all other interleavings.

Proof Rules

for concurrent composition

Modular proof rule for \parallel

$$\frac{p;q \leq r \quad \& \quad p';q' \leq r'}{(p \parallel p');(q \parallel q') \leq (r \parallel r')}$$

- Splits the proof of a complex inequality between concurrent programs into two simpler sequential proofs.
- The proof rule is proved from the exchange law and the exchange law is proved from the rule

Modularity rule implies the Exchange law

$$\frac{p;q \leq r \quad \& \quad p';q' \leq r'}{(p \parallel p');(q \parallel q') \leq r \parallel r'} \quad (\text{modularity rule})$$

- Replace r by $p;q$ and r' by $p';q'$
- The antecedents $(p;q \leq p;q$ and $p';q' \leq p';q')$ are now proved by reflexivity of \leq
- and the conclusion is:

$$(p \parallel p');(q \parallel q') \leq (p;q) \parallel (p';q')$$

which is the exchange law

Exchange law implies Modularity rule

- Assume the antecedents of the rule: $p;q \leq r$ and $p';q' \leq r'$
- $(p;q) \parallel (p';q') \leq (r \parallel r')$ (covariance of \parallel 2ce)
- $(p \parallel p'); (q \parallel q') \leq (p;q) \parallel (p';q')$ (exchange law)
- So $(p \parallel p'); (q \parallel q') \leq (r \parallel r')$ (by transitivity of \leq)
- Therefore $\frac{p;q \leq r \quad \& \quad p';q' \leq r'}{(p \parallel p'); (q \parallel q') \leq r \parallel r'}$ (the modularity rule)

- $$\frac{\frac{\{p\} \ q \ \{r\}}{\{p \parallel p'\}} \quad q \parallel q' \quad \frac{\{p'\} \ q' \ \{r'\}}{\{r \parallel r'\}}}{\{p \parallel p'\} \ q \parallel q' \ \{r \parallel r'\}}$$

Separation Logic
(O'Hearn)

- $$\frac{r \xrightarrow{q} p \quad r' \xrightarrow{q'} p'}{(r \parallel r') \xrightarrow{q \parallel q'} (p \parallel p')}$$

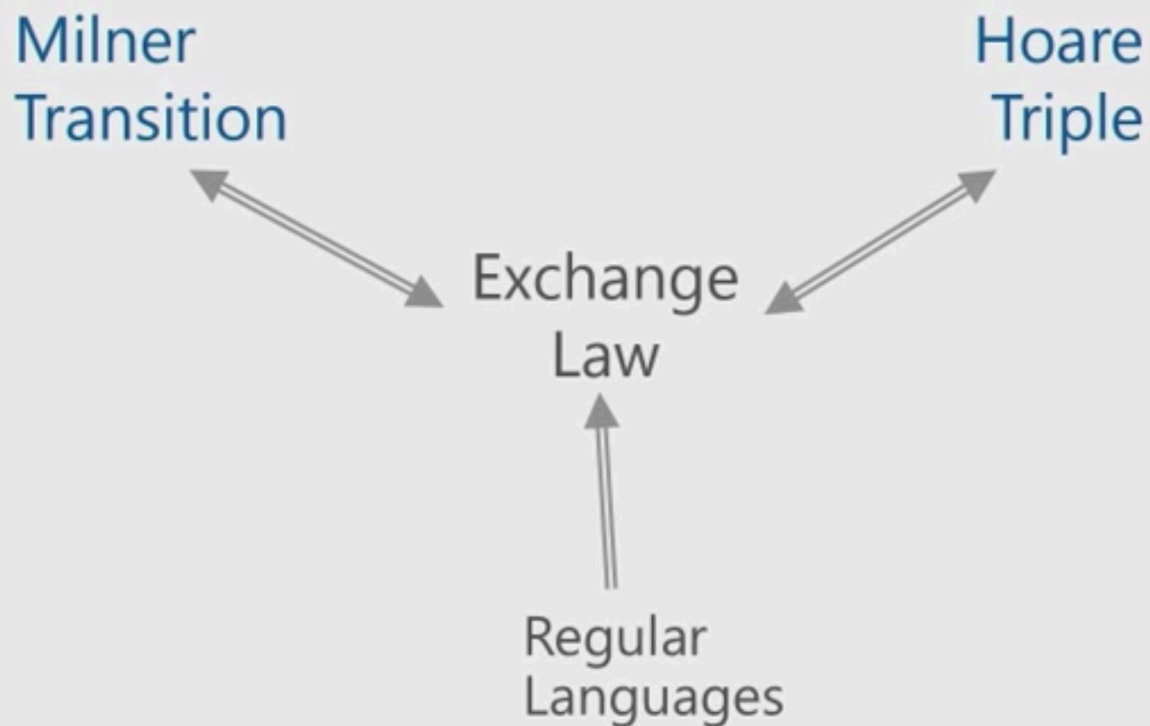
Operational Semantics
(Milner's CCS)

- $$\frac{p; q \leq r \quad p'; q' \leq r'}{(p \parallel p'); (q \parallel q') \leq (r \parallel r')}$$

modularity rule

are all equivalent to the Exchange axiom

Summary: Concurrent Composition





In Praise of Algebra

Algebraic Laws

- are taught to schoolchildren
- appreciated by mathematicians
- used by engineers
- and by software tools
 - optimisers, compilers,
 - program verifiers, generators and analysers,
 - test case generators, displays, and diagnostic aids
- They play a central role in unification of theories

Anybody against?